**Project Notes-2 on**

***House Price Prediction***

***Submitted to***



**Great Learning Olympus**

***Prepared by***

**Mrinmoy Majumdar**

**June-D Batch**

**Post Graduate Program in Data Science & Business Analytics**

**From**

****

UT Austin

**June, 2022**

TABLE OF CONTENT

**1a. Build various models (You can choose to build models for either or all of descriptive, predictive or prescriptive purposes)**……………………………….4

**1b. Test your predictive model against the test set using various appropriate performance metrics………………………………….**……………………….17

**1c. Interpretation of the model(s)……………………………….......................**22

###### **2a. Ensemble modelling, wherever applicable**………………………………...22

**2b. Any other model tuning measures (if applicable) …………...**.....................24

2c. Interpretation of the most optimum model and its implication on the business………...................................................................................................24

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **Images** | **Page No** |
| **Coefficient of linear regression** | **4** |
| **Intercept** | **5** |
| **Training Score** | **5** |
| **VIF** | **5** |
| **Parameter values** | **6** |
| **Linearity plot** | **9** |
| **Density plot** | **9** |
| **Probability plot** | **10** |
| **Scatter plot** | **10** |
| **Regression equation** | **10** |
| **MCE score** | **15** |
| **MCE plot** | **16** |
| **MSE, MAE and RMSE** | **17** |

**LIST OF TABLES**

|  |  |
| --- | --- |
| Table 1 | 6 |
| Table 2 | 7 |
| Table 3 | 25 |
| Table 4 | 25 |

**1)** **Model building and interpretation**

**a) Build various models (You can choose to build models for either or all of descriptive, predictive or prescriptive purposes)**

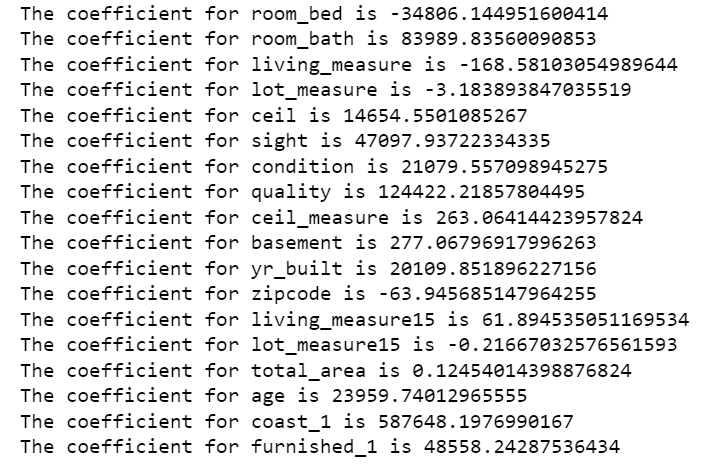
**Model-1: Base Linear Regression**

Before performing model building exercise, we need to split the dataset into training and test sets. The preliminary step is to capture the target column which is price into separate vectors for training and test sets. The ratio for split is 70:30 for training and test samples respectively.

Since it is a regression-based problem, we will start off with linear regression model which can be implemented in 2 ways – either using stats models or basic model package. Assumptions of linear regression also needs to be tested violating of which can lead to severe performance issues.

Copy all the predictor variables into X data frame. Since 'price' is dependent variable drop it. Copy the 'price' column alone into the y data frame. This is the dependent variable. Let us break the X and y data frames into training set and test set. For this we will use scikit learn package's data splitting function which is based on random function. Split X and y into training and test set in 80:30 ratio.

The linear regression function is invoked and the best fit model is found on the training data. Let us explore the coefficients of each of the independent attributes.



**Figure 1: Coefficients for the linear regression model**

Let us check the intercept for the model-



**Figure 2: Intercept for the linear regression model**

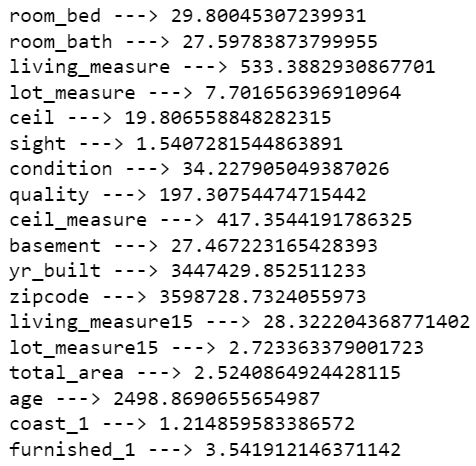


**Figure 3: Training score for the linear regression model**



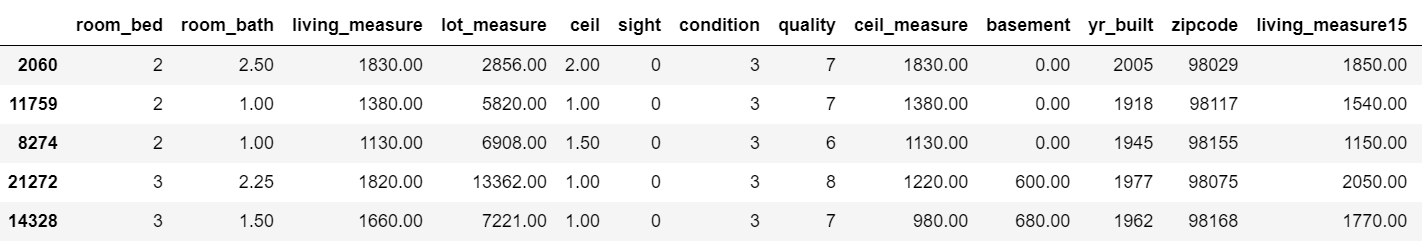
**Figure 4: Testing score for the linear regression model**

So, the base linear regression model explains 65% of the variability in Y using X.



**Figure 5: VIF for the dataset variables**

Columns with VIF > 5 is said to have high multicollinearity which violates one of the assumptions of the linear regression model and hence needs to be dropped before going to the next iteration.

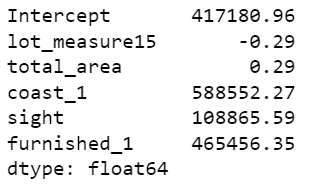


**Table 1: OLS training model head**

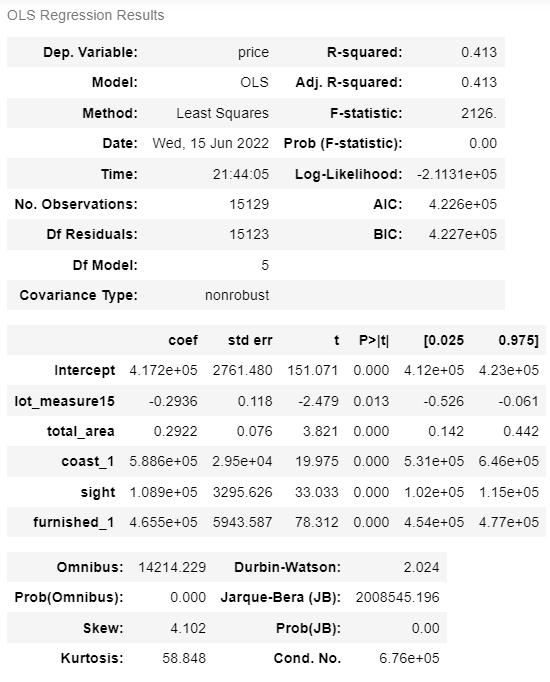
**Model-2: Linear Regression Ordinary Least Square method**

Formula for OLS model -

price ~ lot\_measure15+total\_area+coast\_1+sight+furnished\_1



**Figure 6: Parameter values for OLS model**



**Table 2: OLS summary**

The overall P value is less than alpha, so rejecting H0 and accepting Ha that at least 1 regression co-efficient is not 0. Here all regression co-efficient are not 0.

Inferences –

* Both the R-squared and Adjusted R squared of our model are very low. This is a clear indication that we have been able to create a very good model that is able to explain variance in price of used cars for up to 41%.
* The model is not an underfitting model.
* To be able to make statistical inferences from our model, we will have to test that the linear regression assumptions are followed.

Before we move on to assumption testing, we'll do a quick performance check on the test data.

Observations -

* Root Mean Squared Error of train and test data is identical, indicating that our model is not overfitting the train data.
* Mean Absolute Error indicates that our current model is able to predict house prices within mean error on test data.
* The units of both RMSE and MAE are same. But RMSE is greater than MAE because it penalises the outliers more.

Linear Regression Assumptions -

* No Multicollinearity
* Mean of residuals should be 0
* No Heteroscedasticity
* Linearity of variables
* Normality of error terms

1st assumption has already been checked and treated accordingly.

Mean of residuals is very close to 0. The second assumption is also satisfied.

Homoscedastic - If the residuals are symmetrically distributed across the regression line, then the data is said to homoscedastic.

Heteroscedasticity- - If the residuals are not symmetrically distributed across the regression line, then the data is said to be heteroscedastic. In this case the residuals can form a funnel shape or any other non-symmetrical shape.

We'll use Goldfeldquandt Test to test the following hypothesis

Null hypothesis: Residuals are homoscedastic Alternate hypothesis: Residuals have heteroscedasticity

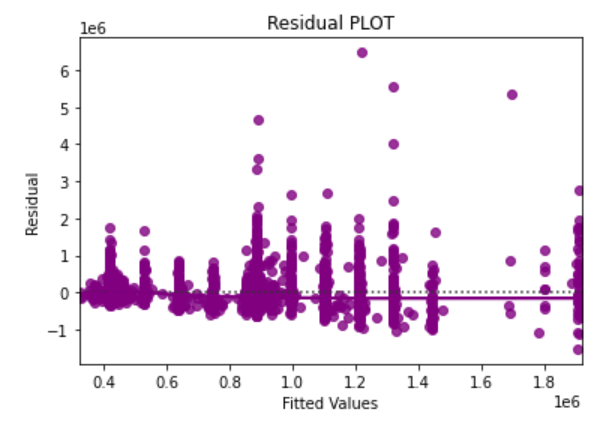
alpha = 0.05

Since p-value > 0.05 we cannot reject the Null Hypothesis that the residuals are homoscedastic.

Assumptions 3 is also satisfied by our model.

Predictor variables must have a linear relation with the dependent variable.

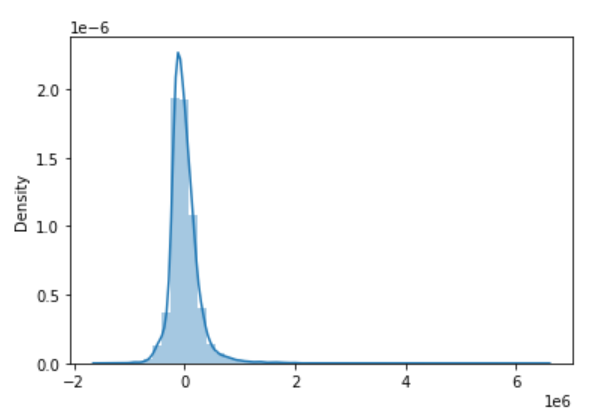
To test the assumption, we'll plot residuals and fitted values on a plot and ensure that residuals do not form a strong pattern. They should be randomly and uniformly scattered on the x axis.



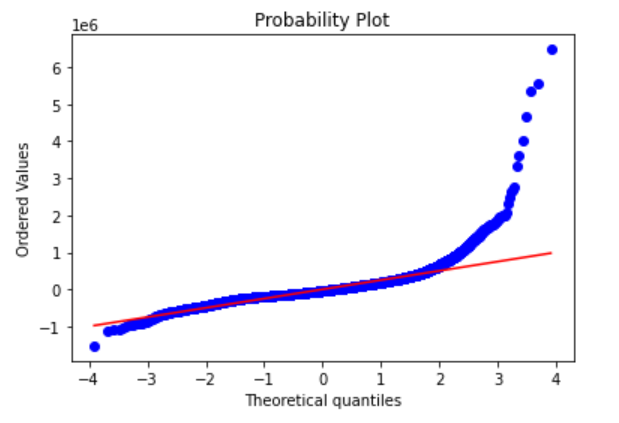
**Figure 7: Plot to check linearity**

Assumptions 4 is satisfied by our model. There is no pattern in the residual vs fitted values plot.

The residuals should be normally distributed.

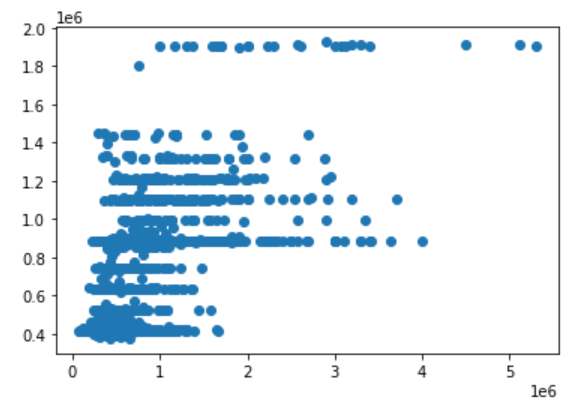


**Figure 8: Residual density plot**

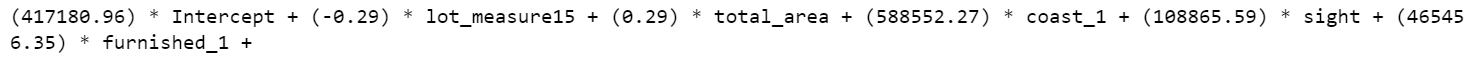


**Figure 9: Residual Q-Q plot**

The residuals have a close to normal distribution. Assumption 5 is also satisfied. We should further investigate these values in the tails where we have made huge residual errors.



**Figure 10: Scatter plot for OLS model**

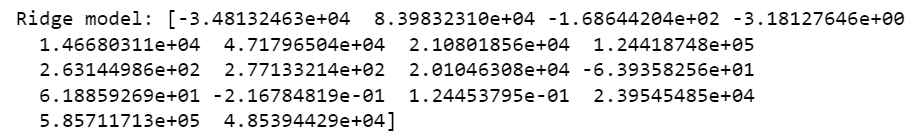


**Figure 11: Regression equation for OLS model**

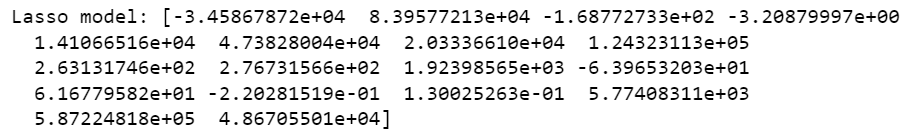
When total area increases by 1 unit, price increases by 0.29 units, keeping all other predictors constant.  
Similarly, when lot measure15 increases by 1 unit, price decreases by 0.29 units, keeping all other predictors constant.

**Model-3: Ridge and Lasso regression**

We will perform ridge and lasso regression to regularize the linear regression model.



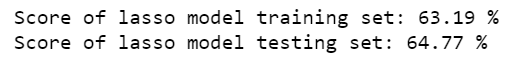
**Figure 12: Ridge model coefficient**



**Figure 13: Lasso model coefficient**



**Figure 14: Ridge model performance**



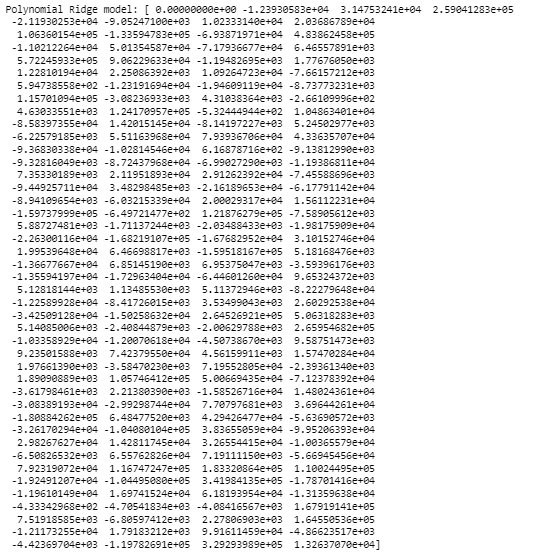
**Figure 15: Lasso model performance**

More or less similar results but with less complex models. Complexity is a function of variables and coefficients. With Lasso, we get equally good result in test and in training. Further, the number of dimensions is much less in LASSO model than ridge or un-regularized model.

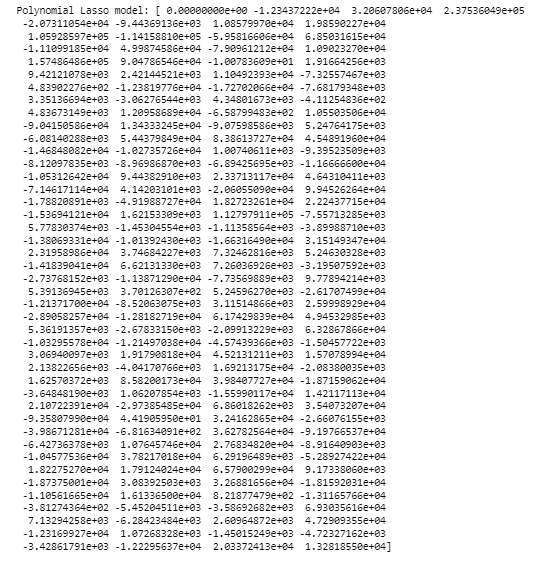
Since on many dimensions, the relationship is not really linear, let us try polynomial models (quadratic). Let us generate polynomial models reflecting the non-linear interaction between some dimensions. Fit a simple non regularized linear model on poly features after scaling the dataset.



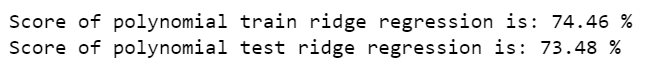
**Figure 16: Coefficient for polynomial regression**



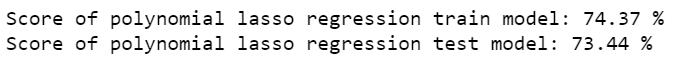
**Figure 17: Coefficient of polynomial ridge model**



**Figure 18: Coefficient of polynomial lasso model**



**Figure 19: Performance of polynomial ridge regression**



**Figure 20: Performance of polynomial lasso regression**

Even with polynomial function, we are getting better results.

**Model-4: Random Forest**

The third model that we look at is random forest which is instantiated using random forest regressor function with maximum depth as 4 since earlier we found 4 clusters were optimum for this problem.

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max features

The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

The scikit-learn implementation combines classifiers by averaging their probabilistic prediction.



**Figure 21: Random Forest train model score**



**Figure 22: Random Forest test model score**

**Model-5: K-Nearest Neighbours**

The next model in line is KNN. For naive bayes algorithm while calculating likelihoods of numerical features it assumes the feature to be normally distributed and then we calculate probability using mean and variance of that feature only and also it assumes that all the predictors are independent to each other. Scale doesn’t matter. Performing a feature scaling in these algorithms may not have much effect.

**Neighbours-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbours of the point.**

Generally, good KNN performance usually requires pre-processing of data to make all variables similarly scaled and centred.

Now let’s apply standard scaler on continuous columns and see the performance for KNN.

The model is built with different values of k. Instantiate learning model with default value of k=5.



**Figure 23: KNN train score for k=5**



**Figure 24: KNN test score for k=5**



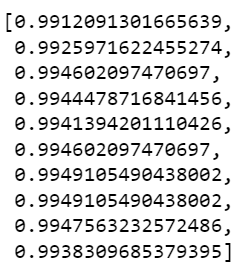
**Figure 25: KNN train score for k=7**



**Figure 26: KNN test score for k=7**

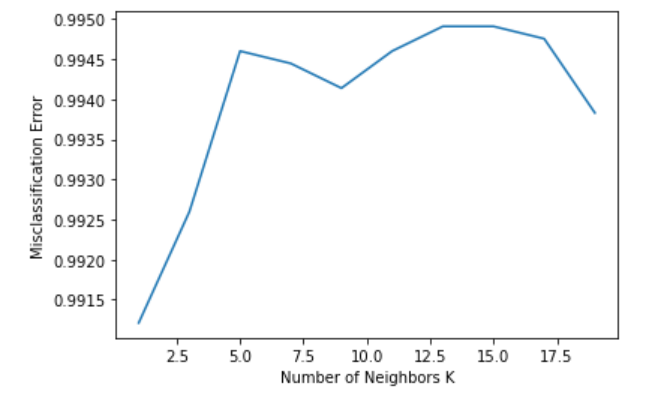
Run the KNN with no of neighbours to be 1, 3, 5 ...19 and find the optimal number of neighbours from the above list using the Misclassification error.

Misclassification error (MCE) = 1 - Test accuracy score. Calculated MCE for each model with neighbours = 1,3,5...19 and find the model with lowest MCE



**Figure 27: MCE scores**

Plot misclassification error vs k (with k value on X-axis) using matplotlib.



**Figure 28: MCE plot**

For K = 1 it is giving the best test accuracy let’s check train and test for K=3 with other evaluation metrics.



**Figure 29: Optimum KNN model training score**



**Figure 30: Optimum KNN model testing score**



**Figure 31: KNN model training score with K=3**



**Figure 32: KNN model testing score with K=3**

**As the difference between train and test accuracies is less than 10%, it is a valid model.**

**Model-6: Support Vector Machine**

**Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier detection.**



**Figure 33: Parameters for SVM model**



**Figure 34: Training score for SVM model 1**



**Figure 35: Testing score for SVM model**

Negative coefficient of determination for both train and set samples indicates poor performance of SVM which can be improved by tuning the model.

**b) Test your predictive model against the test set using various appropriate performance metrics**

Let us check the sum of squared errors by predicting value of y for test cases and subtracting from the actual y for the test cases.



**Figure 36: Training MSE for the linear regression model**



**Figure 37: Testing MSE for the linear regression model**

Under root of mean squared error is standard deviation i.e., average variance between predicted and actual.



**Figure 38: Training RMSE for the linear regression model**



**Figure 39: Testing RMSE for the linear regression model**



**Figure 40: Training MAE for the linear regression model**



**Figure 41: Testing MAE for the linear regression model**

So, there is average of 217661 (roundoff) price difference from real price on an average.

R^2 is not a reliable metric as it always increases with addition of more attributes even if the attributes have no influence on the predicted variable. Instead, we use adjusted R^2 which removes the statistical chance that improves R^2.Scikit does not provide a facility for adjusted R^2, so we use statistical model, a library that gives results similar to what you obtain in R language. This library expects the X and Y to be given in one single data frame.



**Figure 42: MSE of OLS training data**



**Figure 43: MSE of OLS testing data**



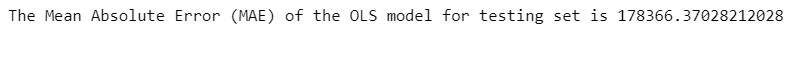
**Figure 44: RMSE of OLS training data**



**Figure 45: RMSE of OLS testing data**



**Figure 46: MAE of OLS training data**



**Figure 47: MAE of OLS testing data**



**Figure 48: MSE of ridge training model**



**Figure 49: MSE of ridge testing model**



**Figure 50: MSE of lasso training model**



**Figure 51: MSE of lasso testing model**



**Figure 52: RMSE of ridge training model**



**Figure 53: RMSE of ridge testing model**



**Figure 54: RMSE of lasso training model**



**Figure 55: RMSE of lasso testing model**



**Figure 56: MAE of ridge training model**



**Figure 57: MAE of ridge testing model**



**Figure 58: MAE of lasso training model**



**Figure 59: MAE of lasso testing model**



**Figure 60: MSE train for polynomial ridge regression**



**Figure 61: MSE test for polynomial ridge regression**



**Figure 62: RMSE train for polynomial ridge regression**



**Figure 63: RMSE test for polynomial ridge regression**



**Figure 64: MAE train for polynomial ridge regression**



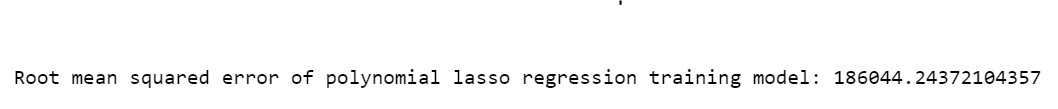
**Figure 65: MAE test for polynomial ridge regression**



**Figure 66: MSE train for polynomial lasso regression**



**Figure 67: MSE test for polynomial lasso regression**



**Figure 68: RMSE train for polynomial lasso regression**



**Figure 69: RMSE test for polynomial lasso regression**



**Figure 70: MAE train for polynomial lasso regression**



**Figure 71: MAE test for polynomial lasso regression**



**Figure 72: Random Forest train RMSE**



**Figure 73: Random Forest test RMSE**



**Figure 74: Random Forest train MSE**



**Figure 75: Random Forest test MSE**



**Figure 76: Random Forest train MAE**



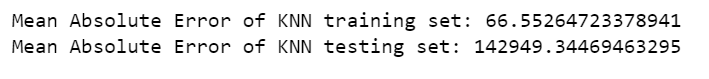
**Figure 77: Random Forest test MAE**



**Figure 78: KNN model RMSE train**



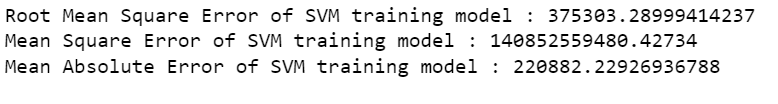
**Figure 79: KNN model RMSE test**



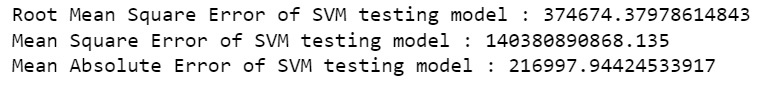
**Figure 80: KNN model MAE train and test**



**Figure 81: KNN model MSE train and test**



**Figure 82: SVM training performance**



**Figure 83: SVM testing performance**

**c) Interpretation of the model(s)**

* Linear regression base model performs better on testing data than training data but the scores are pretty decent.
* KNN model has high accuracy on training and poor scores in testing a clear indication of an overfit model.
* Ridge and lasso regression perform the best among all the models having identical training and testing scores which were improved by leaning towards quadratic models.
* Random Forest scores on training and testing samples prove it to be an average model at best.
* Support Vector Machine is the worst performer of the lot.
* Performance measures are enhanced or decreased according to the model scores.

**2) Model Tuning and business implication**

**a) Ensemble modelling, wherever applicable**

**Ensemble Model-1: Bagging**

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.



**Figure 84: Bagging parameters**

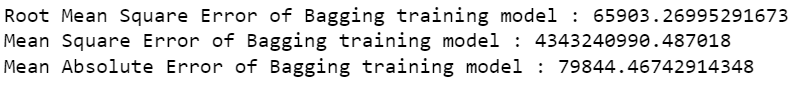


**Figure 85: Bagging training accuracy**

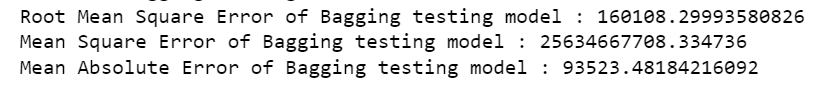


**Figure 86: Bagging testing accuracy**

The model suffers from overfitting as it performs excellently on trained data but not as much on test data.



**Figure 87: Bagging train performance metrics**



**Figure 88: Bagging test performance metrics**

The performance metrics of the model suggests a mixed bag of highs and lows for both types of samples.

**Ensemble Model-2: Ada Boosting**

The module sklearn. ensemble includes the popular boosting algorithm AdaBoost, introduced in 1995 by Freund and Schapire

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.

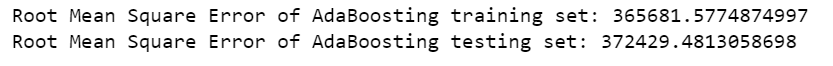
The number of weak learners is controlled by the parameter estimators. The learning rate parameter controls the contribution of the weak learners in the final combination. By default, weak learners are decision stumps. Different weak learners can be specified through the base estimator parameter. The main parameters to tune to obtain good results are estimators and the complexity of the base estimators (e.g., its depth maximum depth or minimum required number of samples to consider a split minimum samples split).



**Figure 89: AdaBoost test score**



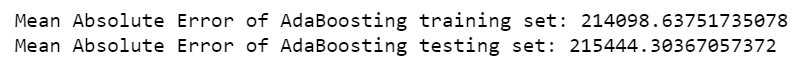
**Figure 90: AdaBoost train score**



**Figure 91: AdaBoost RMSE scores**



**Figure 92: AdaBoost MSE scores**



**Figure 93: AdaBoost MAE scores**

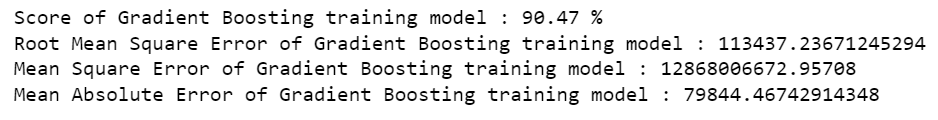
Both the training and testing coefficient of determination shows that the model suffers from underfitting.

**Ensemble Model-3: Gradient Boosting**

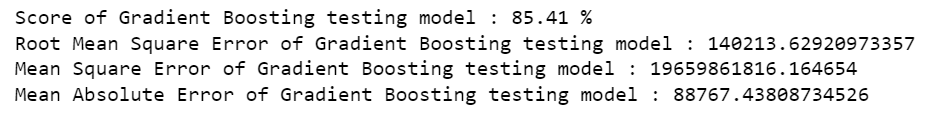
**Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.**



**Figure 94: Gradient Boosting parameters**



**Figure 95: Gradient Boosting training performance**



**Figure 96: Gradient Boosting testing performance**

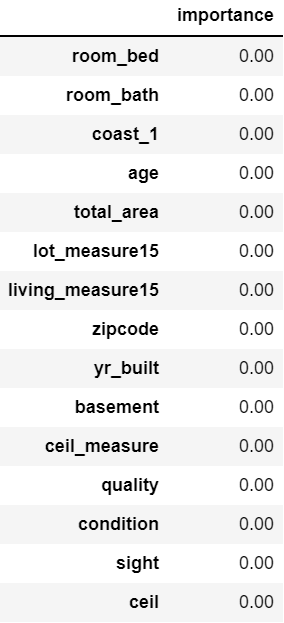
The model performance is the best amongst all the models built till now and can be considered as the final and optimum model for the problem in hand.

**b) Any other model tuning measures (if applicable)**

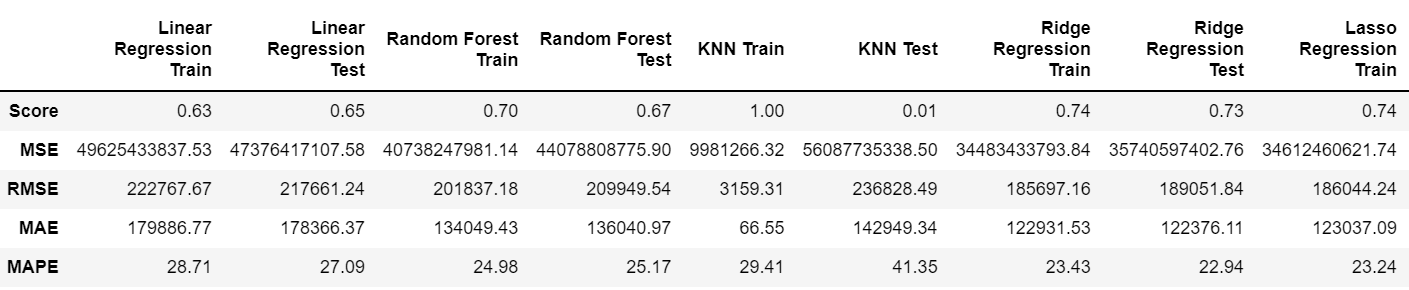
Model tuning measures like hyperparameter tuning or regularizations were not performed on existing base models to improve performance metrics since it is taking a longer time than expected to execute.

**c) Interpretation of the most optimum model and its implication on the business**

The most optimum model selected for house price prediction problem is gradient boosting algorithm. Good performance measures on train and test sets lead to an easy decision to choose the algorithm. We will look at feature importance for the given dataset and accordingly help potential sellers/buyers to quote suitable prices mutually agreed upon by both parties.



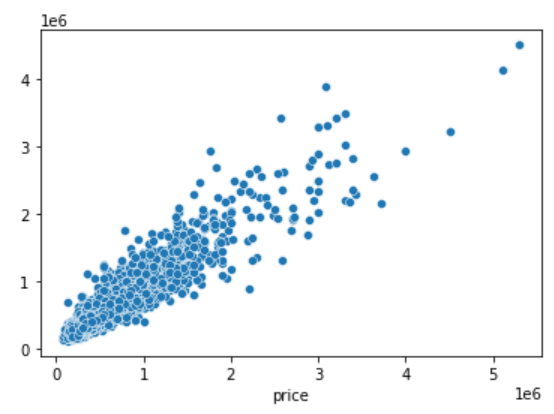
**Table 3: Feature importance of the model**



**Table 4: Model comparison**

Business Insights and Recommendations -

* Our final Gradient Boosting model has a MAPE of 15% on the test data, which means that we are able to predict within 15% of the price value. This is a very good model and we can use this model in production.
* With our linear regression model, we have been able to capture ~85 variation in our data.
* The model indicates that the most significant predictors of price of houses are number of bedrooms, number of bathrooms, age, total area and zip code.
* We will have to analyse the cost side of things before we can talk about profitability in the business. We should gather data regarding that.
* The next step post that would be to cluster different sets of data and see if we should make multiple models for different zip code/house age.
* Bedroom and bathroom presence is the basic requirement for any customer looking for a house hence justifiably it tops the in-demand features.
* Houses present near the coast is a priority surprisingly hence real estate businesses should target potential sellers present in this location.
* Condition and quality of the house languishes at the bottom therefore customers are not considering other’s feedback as expected.



**Figure 97: Actual vs predicted price for optimum model**